Key Management for Onion Routing in a True Peer to Peer Setting

Paolo Palmieri[1] and Johan Pouwelse[2]

Parallel and Distributed Systems, Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands
[1]p.palmieri@tudelft.nl [2]peer2peer@gmail.com

Abstract. Onion routing is a technique for anonymous and privacy preserving communication at the base of popular Internet anonymity tools such as Tor. In onion routing, traffic is relayed by a number of intermediary nodes (called relays) before it reaches the intended destination. To guarantee privacy and prevent tampering, each packet is encrypted multiple times in a layered manner, using the public keys of the relays. Therefore, this mechanism makes two important assumptions: first, that the relays are able to communicate with each other; second, that the user knows the list of available relays and their respective public keys. Tor implements therefore a distributed directory listing the relays and their keys. When a user is not able to communicate with relays directly, he has to use special bridge servers to connect to the onion network.
This construction, however, does not work in a fully peer to peer setting, where each peer only knows a limited number of other peers and may not be able to communicate with some of them due, for instance, to NAT or firewalls. In this paper we propose a key management scheme for onion routing that overcomes these problems. The proposed solution does not need a directory system and does not imply knowledge of all active relays, while it guarantees the secure distribution of public keys. We also present an alternative strategy for building circuit of relays based on bloom filters. The proposed construction overcomes some of the structural inefficiencies of the Tor design, and opens the way for implementing onion routing over a true peer to peer overlay network.

Keywords: Key Management, Onion Routing, Peer to Peer

1 Introduction

At the beginning of its diffusion to the general public, Internet was perceived by most users as a private or even anonymous medium of communication [19]. However, this perception has changed over the years, and the lack of online privacy is becoming more and more evident (and for some worrisome) to the users. Anonymous communication over the Internet is thus receiving increased interest and attention. A particular kind of application whose users are particularly concerned about privacy is that of peer to peer (p2p) systems. However, most peer to peer applications currently lack effective measures to protect the privacy of their

users. Those that do attempt at building privacy-preserving and anonymous p2p networks, such as Freenet [6] or GNUnet [3, 2], have often found scarce adoption, mostly due to the relatively difficult operation of such software by inexperienced users and (paradoxically) to the small existing user-base, which makes them unattractive to new users. The first software for private and censorship-resistant communication to receive widespread attention by non specialized users was Tor (The Onion Router) [7], who now counts millions of active users. Tor can be used to connect to existing Internet services in an anonymous manner, but it is not designed for peer to peer: users are actually actively discouraged from running p2p software such as Bittorrent over its network. Likewise most tools for achieving anonymity over the Internet, Tor relies on a multiple relay setup, in which communication between the originator (the user) and the destination (the web service) is tunneled through a number of proxies, called relays. Tor uses a specific approach to the relaying of messages, called Onion Routing (OR) [20]. In onion routing, messages are repeatedly encrypted: each relay removes a layer of encryption, and then forwards the message to the next relay, where this is repeated. This prevents intermediary hops from learning the contents, origin and destination of any message that passes through them. In the Tor setup, relays are publicly disclosed and organized in a directory system, which allows users to locate active relays and retrieve their public key information. The directory is especially needed for one reason: clients communicate only with the first relay in a circuit, and rely on its services to talk to relays further down the chain in order to preserve their privacy. This prevents them from contacting directly other relays to obtain their public key, and makes the directory necessary to prevent the first node from impersonating other relays in the circuit. However, while the directory is distributed among several "trusted" nodes, this scenario is far from a real peer to peer setting, in which we can not assume peers to have a full view of the network, or even be able to connect to all nodes (due to NATs, firewalls, etc.). Therefore, if we are to implement an onion routing mechanism over a true peer to peer setting, we need a key distribution and management mechanism that takes into account the critical peculiarities of peer to peer systems.

1.1 Contribution

In this paper, we discuss why the directory infrastructure used by Tor for key distribution is not applicable to a peer to peer setting, where peers are at the same time clients and relays, they do not know the full list of other relays or the network topology and may not even be able to connect to part of the peer list known to them. While key management schemes for peer to peer settings already exist in literature, none of them addresses the specific needs of distributed onion routing systems, and in particular the impossibility of challenging peers directly to confirm their identity or obtain their keys. Existing peer to peer designs based on anonymizing relays, on the other hand, do not implement a Tor-like onion routing, which, due to the resilience to attacks proved by the Tor network, is believed to be a reliable privacy-preserving mechanism. We address these issues by proposing a novel scheme implementing a Tor-like secure key distribution

and management mechanism based on a true peer to peer overlay. The solution presented in this paper allows the secure retrieval of temporary onion keys, and their validation against the long-term identity key for each relay, while not requiring a directory structure based on trusted known nodes. In the design of our solution we also provide an innovative mechanism for selecting relays during circuit creation based on bloom filters, which might be of independent interest.

1.2 Related works

While no established peer to peer software uses onion routing, its adoption has been proposed in various theoretical designs [14]. However, the lack of a distributed Tor-like key management solution imposed to deviate from the standard onion routing paradigm. ShadowWalker [15] constructs circuits based on a random walk over a redundant structured topology, while Saboori and Mohammadi propose a design relying on supernodes so that no peer can identify the communicating parties with certainty [18]. Landsiedel et al. instead extend onion routing by allowing the first half of the path to be selected by the sender and the second half by the receiver of the packet [9]. A sizable part of relevant literature focuses instead on a way to perform secure and anonymous lookups (in this context, useful for retrieving the public keys) [21]. Torsk [12], in particular, utilizes a combination of two p2p lookup mechanisms, in order to preserve the confidentiality and integrity of lookups. On the other hand, distributed key management schemes based on the concept of distributed hash table have been proposed in the past [17, 22, 11]. These mechanisms present solutions for a number of different purposes: among them, multimedia streaming services, where the goal of distributed key management is to safeguard content security [17, 16], mobile ad-hoc networks [13], or (heterogeneous) wireless sensor networks [10, 5].

2 Onion Routing and Key Management in Tor

The concept of onion routing was first proposed by Syverson, Goldschlag and Reed in 1997 [20], but it was only in 2002 that Dingledine, Mathewson and again Syverson refined the paradigm and finally implemented it as the Tor software [7]. The aim of onion routing is to protect the identity and network activity of its users from surveillance and traffic analysis. This goal is achieved by concealing to external observers both the content and routing information of the user's traffic, which is forwarded through a virtual circuit composed of three successive relays. Each relay only knows the preceding and following node: this guarantees sender's anonymity to all relays except the first and the secrecy of the destination to all relays except the last. At the same time, communication is repeatedly encrypted with a public key encryption scheme in a layered manner, using in inverse order the public keys of all the relays in the circuit. Traffic going back to the user from the destination is similarly encrypted and routed from the last to the first relay, and on to the user. In general, Tor allows users to connect to any Internet service independently of the protocol, as long as the software supports Socks proxying.

Tor uses a number of different encryption keys, for three different purposes: encryption of traffic, verifying the identity of the relays, and making sure that all clients know the same set of relays. Encryption is performed on all connections within the Tor network. Tor uses TLS link encryption, which guarantees that observers can not learn which circuit a given packet is intended for. Furthermore, Tor clients establish a session key with each relay in the circuit. This extra layer of encryption ensures that only the exit relay can read outbound packets, and only the users can decrypt inbound traffic. Both sides discard the session keys when the circuit is disposed of, achieving forward secrecy. Every Tor relay generates regularly (once a week) a public and private key pair called *onion key*. When a Tor client establishes a circuit, at each step it challenges the Tor relay to prove knowledge of its onion key, to prevent spoofing. Since the Tor client chooses the path independently, circuits are based on the concept of *distributed trust*: no single relay can learn both the client address and the traffic destination. The list of relays and their keys are shared through a distributed *directory*. Fast and stable servers are selected for the task directly by the Tor developers, and locations and public keys for each directory authority (a relay hosting the directory service) are hard-coded into the Tor client software. Each relay has a long-term signing key pair called *identity key*. Each directory authority additionally has a *directory key*. The directory authorities provide a list of all known relays signed with its directory key. The list also contains the onion key for each relay (signed with their identity key).

3 Distributed Hash Table (DHT) Networks

Large decentralized, distributed systems are notoriously difficult to design and implement. Successful designs are often based on simple primitives, such as the *distributed hash table* (DHT). A DHT can be used to efficiently store and retrieve data (for instance, files) in a distributed manner, while respecting the three main properties of peer to peer systems: operating without central coordination; being able to accommodate nodes joining, leaving, or failing without disruption of service; and functioning efficiently for any number of peers. Over the years, many different DHT's have been proposed in literature, but only a subset has found widespread usage. A notable example is Mainline DHT, introduced by the developers of peer to peer software Azureus and then incorporated in a slightly modified version in the BitTorrent protocol, used daily by millions of users. In general, DHT are nowadays regarded as one of the most stable and efficient setups for peer to peer network [1]. Given the growing interest in privacy preserving techniques for DHT networks [8], we base our scheme on this network structure.

A DHT is built over a *keyspace*[1], usually defined by a hash function: an example is the set of all strings of length 160 bits output by SHA-1. A partitioning scheme for the keyspace is used to divide responsibility for the keyspace among

[1] We use the term keyspace for consistency with the relevant literature, bit it is important to clarify that the keyspace of a DHT has no relation to the keyspace of the PKC schemes we use for the purposes of onion routing in the following.

the peers participating in the network. This is achieved by defining function measuring the *distance* between two values in the keyspace, and by assigning to each peer a value in the keyspace called *identifier*: the peer is then responsible for the subset of values that have less than a predefined distance from his identifier value. Finally, an overlay network connecting the peers is generated. Each peer maintains a connection to a number of other peers, called *neighbors*. Neighbors are selected according to a certain structure, known as the network topology, which, together with the hash and distance functions, defines the specific DHT. Once a DHT is established, peers are able to find peers responsible for any given value in the keyspace. While the design of our key management scheme is independent of the specific DHT, we assume the DHT to specify: a keyspace K of size s; a hash functions h mapping information that can be retrieved through the network (files, etc.) to values in the keyspace; a function f_{dist} for determining the distance between two values of the keyspace; and the distance d within which a peer is responsible for keys in the keyspace. We also assume the keyspace to be two-dimensional: given a distance d, a function f_{dist}, and a value v in the keyspace, there are exactly two values v^+ and v^- for which

$$f_{dist}\left(v^+\right) = f_{dist}\left(v^-\right) = d \ . \tag{1}$$

The distance d should be calibrated over the number of peers in the network: if the value is too high, each peer has to store too much information, while if d is too low information is spread too thin and may be difficult or impossible to retrieve. For this reason, some DHT constructions adjust this value during operation of the network. The key management scheme for onion routing we propose in the following is therefore able to accommodate for a varying d.

4 Onion Routing Key Management over a DHT Network

In the following we propose a key management mechanism for achieving onion routing over an existing DHT peer to peer network. In particular, we imagine a scenario in which peers in the network communicate and transmit information according to the employed DHT construction, but also want to preserve their privacy by using onion routing in the communication within the peer to peer overlay. Therefore, when retrieving information (such as files) over the peer to peer network, peers relay the communication through a number of other peers, following the onion routing principles described in Section 2.

Being a peer to peer scenario, we assume that each peer in the network acts as both a regular user and as a relay, as opposed to the Tor implementation in which most relays are dedicated servers and most users do not relay traffic. In order to be able to act as a relay, a peer creates during bootstrap an *identity key* pair, using a public key encryption scheme. Identity keys, similarly to the key management of Tor, are kept by the peers indefinitely. Additionally, the peer also creates a temporary *onion key* pair, which expires after a predetermined amount of time and is then replaced by a new pair. The two different key pairs serve different purposes: the identity key is used prove the identity of the peer

and sign information regarding it, so that other peers can verify it as legitimate. The onion key, instead, is used for actual communication during the creation of onion circuits, and specifically before the generation of the symmetric session key with which the peer encrypts traffic for the relays and vice versa. In order to participate in the network, a peer needs to distribute the public keys of both his identity and onion pairs. In the following, we describe how the different keys are spread across the network, and which peers are responsible for storing and distributing the keys. We say that a peer *owns* a key (whether it is an identity or an onion key) if he generated the key himself, while we say that a peer is *responsible* for a key if he stores the key and distributes it to other peers requesting it. The novel key distribution scheme we propose still relies on the underlying DHT structure for the actual transmission of the information: keys are transmitted over the network in the same way files are, following the specific DHT protocol employed. We assume the number of relays in a circuit to be 3.

Identity Keys Distribution The peers responsible for the identity key of a peer are determined by hashing the identifier for the peer using the hash function employed by the DHT scheme, similarly to any information shared in the network. As defined in the previous section, a peer X is responsible for the identity key $i_P \in K$ of the peer P if: $f_{dist}\left(X - h\left(P\right)\right) < d$. This defines a subset I_B of size $2d$ of the keyspace K. The peer owning the key is responsible for initiating the distribution of its identity key to the peers responsible for storing it. This is done using the information retrieval algorithm of the underlying DHT scheme. Should d increase during operation, peers already responsible for the key distribute it to the new peers in I_P. Should d decrease, instead, peers that are no longer in I_P just drop the key and stop distributing it.

Onion Keys Lifetime Similarly to the Tor design, we require onion keys to be replaced after a number of hours t. Tor uses a time interval of one week, but in our peer to peer design we leave the decision on the interval duration to the protocol implementation, depending on the specific needs of the network. In order to prevent flooding of the network at the end of each validity interval, when all onion keys expire at the same time, we divide the peers into subgroups, and we assign to each subgroup a different time offset from the reference expiration time. We determine the time offset according to the identifier of the peer owning the onion key: we consider the first n bits of the identifier value in order to have 2^n different expiration times. For instance, given a reference time, all peers with the first n bits of the identifier having value 0 have no offset (their key expires at the reference time), those with value 1 have a positive offset of one hour (their key expires one hour after the reference time) and so on. Assuming identifiers are chosen uniformly in the keyspace, this divides equally the peers. Moreover, if the identifier of a peer is known, so are the expiration times of his onion keys.

Onion Keys Distribution Temporary onion keys are stored within the DHT network similarly to identity keys. However, peers responsible for storing the identity key of a peer will never be responsible for his onion keys too. Moreover, a peer responsible for the onion key of a peer for the current time interval,

will not be responsible for a key of the same peer for a number of following time intervals. We achieve this using the following distribution scheme. For each peer P and his identifier $i_P \in K$, we divide the keyspace (and hence the other peers in the network) into $u = \frac{s}{2d}$ partitions of size $2d$, such that one such partition coincides with I_P, as defined above. Peers in each partition except I_P cyclically store the current onion key for P. The arbitrary function

$$f_{on} : \ K \rightarrow \{\text{all possible partitions of } K \text{ of size } 2d\}, \tag{2}$$

defines the partitions. It takes an identifier $i \in K$ as input and outputs the set of partitions $\{O_1, \ldots, O_{u-1}\}$, such that $O_1, \ldots, O_{u-1}$ are disjoint subsets of K of equal size and $O_1 \cup \ldots \cup O_{u-1} = K \setminus \{I_P\}$. In any $u-1$ contiguous time intervals, each available partition is selected once for storing the onion key of the peer, starting with O_1 and proceeding in ascending order till O_{u-1}. The peer owning the key is responsible for the distribution of the next key to the peers in the appropriate partition O_n before the current key expires. Should d change, the partitions are adjusted from the next time interval. When a peer is queried for onion keys he is responsible for, he always reply with all of them: that is, queries are not made for retrieving a specific key, but for retrieving all keys under the responsibility of one peer.

4.1 Building an Onion Circuit

The process of building an onion circuit is more complex in a peer to peer setting than it is for Tor, as peers have only a partial view of the network. In building a circuit, this means that two consecutive relays may not able to communicate with each other, thus causing the circuit to fail. This issue may be addressed by letting the first relay in a circuit decide the second, and so on. However, if the user stumbles on a first relay that is malicious, then the whole circuit will be likely composed of malicious nodes. Therefore, we decide to keep the circuit creation and relay selection under the user's direct control, and we propose the use of bloom filters to let the user identify which relays can connect to each other.[2] In particular, we require each peer to attach to his public onion key a bloom filter where the neighbors of the peer have been encoded, called *neighbor bloom filter* and built over the neighbors' identifiers. The filter, as well as the onion key, must be signed by the peer with his identity key, and is stored and distributed together with the onion key. The neighbor filter lets us construct circuits in which the relays can communicate to each other: the peer building the circuit can in fact calculate the intersection of the filters of the potential first and last relays to identify whether they have a common neighbor that can be used as middle relay. But the bloom filter does more than that: it also allows us to evaluate the relatedness of relays, based on the number of common neighbors.

[2] Bloom filters (BF) are space-efficient data structures representing a set. A BF generated for a set allows to determine, without knowledge of the set itself, whether an element is in the set or not. In the following we assume knowledge of the bloom filter definition and properties: the interested reader can find a thorough discussion in [4].

We may want, for instance, to build a circuit in which the first and third relays are not neighbors, nor have more than n common neighbors, in order to diversify the circuit and reduce its locality (for a security discussion, see Section 5).

Considering that all peers are responsible, at some point in time, for an onion (or identity) key for all other peers, due to the distribution mechanism described above, all peers are naturally aware of most of other peers (the exception being the peers that entered the network later than $u - 1$ time intervals before the current one). No further step is therefore necessary for a peer to discover new peers prior to building a circuit, as the identifiers of (most) other peers are known. A circuit is built as follows:

1. The peer selects a potential first relay among his neighbors, and acquires his onion key and neighbor filter through the DHT (and not directly from the neighbor himself).
2. Once the relay information for the first relay has been acquired, the peer identifies potential third relays by calculating the intersection between their neighbor filter b_3 and the one of the first relay b_1. The peer queries peers through the DHT to acquire additional relay information if necessary. Then he selects a third relay satisfying the following requirements: not being a neighbor of the first one nor the peer; and having a number of common neighbors with the first relay within the range defined as acceptable by the peer. Once the third relay is selected, the peer sends to the first relay the intersection filter $b_i = b_1 \cap b_3$. Then, the peer instructs the first relay to build a circuit with one of his own neighbors whose identifier satisfies b_i.
3. The peer communicates to the second relay through the first one in an encrypted manner (following the onion routing scheme), and instructs him to extend the circuit to the selected third relay. The second relay is a neighbor of both the first and third relay minus the false positive probability of the intersection bloom filter. In case the second relay is unable to communicate to the third one, the peer starts again from step 2.

5 Security Analysis

In this section we analyze the security of the proposed key management scheme.

Distributed Trust No single relay in the circuit should be aware of both the initiating peer and the traffic destination. We achieve this by letting the peer select independently the first and last relay, and by not disclosing to the first relay the identity of the third one. This is possible thanks to the use of the intersection of neighbor filters, as explained in step 2 of the circuit building protocol. This, in turn, allows the first relay to extend the circuit to a neighbor able to communicate with the third relay without knowledge of his identity. It is important to note here that the first relay does learn the list of potential third relays, as this is the same list as the list of neighbors of the second relays. However, since relays can only communicate with their neighbors, this issue is impossible to prevent in a peer to peer setting.

Information Leakage when Querying Relay Information When a peer queries other relays through the DHT for retrieving relay information, he somehow reveals his willingness to build a circuit with one of those relays. Calibrating the onion key lifetime and increasing the number of queries performed by a peer during each time interval is however sufficient to prevent potentially identifying information from being leaked: computing a correlation is possible only if the queries recover relay information for a limited number of peers only.

Identity Key Verification The identity key of each peer is verified directly by the peers responsible for its distribution through a challenge. When the challenge can not be accomplished (for example, if the peer responsible for the key is unable to contact the owner), the peer responsible for the key does not distribute that key. Having a subset of peers distributing the identity keys makes an attack possible if at least a majority of those peers collude in order to reply with a fake key instead of the real one. However, we note here that the peer building the circuit obtains onion keys from a different set of peers, and is also aware of the partition of peers in the network responsible for that specific identity key. The attack is therefore limited to preventing a peer from using the selected peer as relay, as the onion key can not be verified against the (fake) identity key. Moreover, should a subset of peers be responsible for a statistically relevant number of un-verifying keys, the peer building the circuit can choose to discard keys from that subset entirely.

Onion Key Verification The onion key for the current time interval is verified by the peer directly with the relay through the circuit, by asking the relay to reply a challenge in the same way Tor does.

6 Conclusions

In this paper, we propose a key management scheme for the distribution of onion keys over a DHT peer to peer network. Achieving onion routing in a true peer to peer setting is in fact a complex task, as no participating peer can be assumed to be able to connect to any other peer. This, in turn, makes the creation of a distributed relay directory similar to the one implemented by Tor impossible. The key distribution scheme we propose addresses this problem by distributing the onion encryption keys among the peers in an decentralized manner. We enable peers to select circuits without knowledge of the full list of relays, and we ensure that selected relays are able to communicate with each other without disclosing their identity by using a bloom filter structure indexing each relay's neighbors. The proposed construction allows the implementation of onion routing over peer to peer application used every day by millions of users, and finally provides and answer to these user's increasing demand for privacy.

References

1. Balakrishnan, H., Kaashoek, M.F., Karger, D.R., Morris, R., Stoica, I.: Looking up data in p2p systems. Commun. ACM 46(2), 43–48 (2003)

2. Bennett, K., Grothoff, C., Horozov, T., Patrascu, I.: Efficient sharing of encrypted data. In: Batten, L.M., Seberry, J. (eds.) ACISP. LNCS, vol. 2384, pp. 107–120. Springer (2002)
3. Bennett, K., Grothoff, C., Horozov, T., Patrascu, I., Stef, T.: The GNet whitepaper. Tech. rep., Purdue University (2002)
4. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM 13(7), 422–426 (1970)
5. Chung, K.I., Sohn, K., Yung, M. (eds.): 9th International Workshop on Information Security Applications, WISA 2008, LNCS, vol. 5379. Springer (2009)
6. Clarke, I., Sandberg, O., Wiley, B., Hong, T.W.: Freenet: A distributed anonymous information storage and retrieval system. In: Workshop on Design Issues in Anonymity and Unobservability. LNCS, vol. 2009, pp. 46–66. Springer (2000)
7. Dingledine, R., Mathewson, N., Syverson, P.F.: Tor: The second-generation onion router. In: USENIX Security Symposium. pp. 303–320. USENIX (2004)
8. Isdal, T., Piatek, M., Krishnamurthy, A., Anderson, T.E.: Privacy-preserving p2p data sharing with oneswarm. In: SIGCOMM. pp. 111–122. ACM (2010)
9. Landsiedel, O., Pimenidis, L., Wehrle, K., Niedermayer, H., Carle, G.: Dynamic multipath onion routing in anonymous peer-to-peer overlay networks. In: GLOBECOM. pp. 64–69. IEEE (2007)
10. Lu, K., Qian, Y., Guizani, M., Chen, H.H.: A framework for a distributed key management scheme in heterogeneous wireless sensor networks. IEEE Transactions on Wireless Communications 7(2), 639–647 (2008)
11. Luo, Z., Li, Z., Cai, B.: A self-organized public-key certificate system in p2p network. Journal of Networks 6(10), 1437–1443 (October 2011)
12. McLachlan, J., Tran, A., Hopper, N., Kim, Y.: Scalable onion routing with torsk. In: ACM CCS. pp. 590–599 (2009)
13. van der Merwe, J., Dawoud, D.S., McDonald, S.: A survey on peer-to-peer key management for mobile ad hoc networks. ACM Comput. Surv. 39(1) (2007)
14. Michéle, B.: Using Onion Routing in Well-Established P2P Networks to Provide Anonymity. Master's thesis, Technische Universität Berlin (December 2008)
15. Mittal, P., Borisov, N.: Shadowwalker: peer-to-peer anonymous communication using redundant structured topologies. In: ACM CCS. pp. 161–172 (2009)
16. Naranjo, J.A.M., López-Ramos, J.A., Casado, L.G.: Key management schemes for peer-to-peer multimedia streaming overlay networks. In: WISTP. LNCS, vol. 5746, pp. 128–142. Springer (2009)
17. Qiu, F., Lin, C., Yin, H.: EKM: An efficient key management scheme for large-scale peer-to-peer media streaming. In: PCM. LNCS, vol. 4261, pp. 395–404. Springer (2006)
18. Saboori, E., Mohammadi, S.: Anonymous communication in peer-to-peer networks for providing more privacy and security. CoRR abs/1208.3192 (2012)
19. Sheehan, K.: Toward a typology of internet users and online privacy concerns. Inf. Soc. 18(1), 21–32 (2002)
20. Syverson, P.F., Goldschlag, D.M., Reed, M.G.: Anonymous connections and onion routing. In: IEEE Symposium on Security and Privacy. pp. 44–54. IEEE (1997)
21. Wang, Q., Mittal, P., Borisov, N.: In search of an anonymous and secure lookup: attacks on structured peer-to-peer anonymous communication systems. In: ACM CCS. pp. 308–318. ACM (2010)
22. Wen, Z., zhang Niu, S., cheng Zou, J.: A Key Management Mechanism for DHT Networks. In: IIH-MSP. pp. 339–342. IEEE (2012)